Trajectory planning under time-constrained communication

Yaroslav Marchukov and Luis Montano*

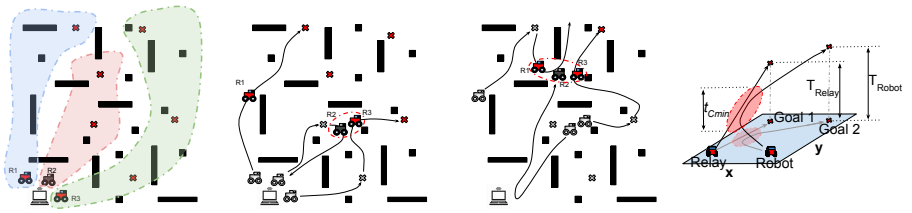
Instituto Universitatio de Investigación en Ingeniería de Aragón, University of Zaragoza, Spain
{yamar,montano}@unizar.es

Abstract. In the present paper we address the problem of trajectory planning for cooperative scenarios in which some robots has to exchange information with other moving robots for at least a certain time, determined by the amount of information. We are particularly focused on scenarios where a team of robots must be deployed, reaching some locations to make observations of the environment. The information gathered by all the robots must be shared with an operation center (OP), thus some robots are devoted to retransmit to the OP the data of their teammates. We develop a trajectory planning method called Time-Constrained RRT (TC-RRT). It computes trajectories to reach the assigned primary goals, but subjected to the constraint determined by the need of communicating with another robot acting as moving relay, just during the time it takes to fulfill the data exchange. Against other methods in the literature, using this method it is not needed a task allocator to assign beforehand specific meeting points or areas for communication exchange, because the planner find the best area to do it, simultaneously minimizing the time to reach the goal. Evaluation and limitations of the technique are presented for different system parameters.

1 Introduction

Let consider a scenario where a team of robots must be deployed to inspect an environment. The mission of the robots is to reach some points of interest and make observations of the environment (taking pictures, environmental sensing...). The information must be transmitted to a static operation center (OP), where human operators monitor the mission. The robots and the OP have a limited range of communications, so they cannot be continuously communicated. Each robot is equipped with a wireless antenna, thus it creates a dynamic communication area, allowing to extend during the motion the communication area determined by the OP. This fact can be exploited by the system to exchange information between the robots. The exchange can be made in different ways. The

* This work was partially supported by Spanish projects DPI2012-32100, DPI2016-76676-R-AEI/FEDER-UE and T04-DGA-FSE.



(a) (b) (c) (d)

Fig. 1. In (a), 3 robots must make observations of the environment (red crosses). The OP allocates the tasks to each robot (the colored areas associated to each in the Fig.(a). Additionally R2 (black) is assigned to communicate directly with OP. Each robot computes its trajectory, sharing it with the mates. Fig.(b)-(c) represent the trajectories planned by R1 and R3 to meet R2 and share data when they make observations of the environment. Fig.(d) illustrates a simplified example for two robots in the x-y-$time$ space, in which Relay is assigned for communications while moving towards its assigned goal, and Robot for primary tasks. The latter computes its trajectory towards its goal, constrained by the time needed for information exchange ($t_{c_{min}}$).

simplest one, and probably the costliest in terms of time and resources, is that robots periodically visit the OP. For large scenarios this solution wastes a lot of time only on information exchange. Another option is to devote some robots to relay tasks, re-transmitting the information of the mates to the OP. However, in a multi-robot application, minimizing the number of these relay robots, maximizes the number of robots visiting the allocated primary tasks. Therefore, a better solution, is to allocate the nearest goals to some robot, which is responsible of visiting these goals and, at the same time, to communicate directly with the OP. This way, the robots devoted to reach farthest goals, share the collected data with the relay robot, which will upload this information to the OP when will go to share its own data. In the same way, the OP updates the mission tasks and this information is shared with the team, through the relay robot. This procedure is illustrated in Fig.1(a)-1(c).

Furthermore, a constant connectivity between robots, requires coordination, considerably constraining the motion. Nevertheless, based on the amount of collected data and the achievable bit-rate, the robot knows the time required to exchange the information. So, it can plan the trajectory, synchronizing with the relay only the required time to fulfill the transmission. As a result, the trajectory of the robot suffers the minimum deviation from the primary goal, at most the time needed to exchange information with the relay robot, see Fig.1(d).

We assume that a task allocation algorithm has assigned the the sequence of goals and the role of the robots, some of them for both, primary and relay tasks. We focus this work on developing a Time-Constrained trajectory planner based on a Rapidly exploring Random Tree algorithm (RRT), called TC-RRT. It computes a trajectory which leads the robot to a primary goal through the communication area of the relay robot. The planner attempts to reduce the time of the entire trajectory, remaining synchronized with the relay the time needed to complete the transmission.

The advantages of the presented algorithm are:

- the planner does not require a task allocator to define beforehand meeting points or areas for communication: itself computes the time-constrained trajectory towards its assigned primary goal.
- it also obtains the best area in which the exchange can be made, so the trajectory is obtained through that area.
- the time for communication can be adapted to the requirements of the amount of collected data for each application and the employed communication technology. The robot is able to find out the time required for each specific transmission, applying it to the best trajectory computation.

The rest of the paper is organized as follows: in Sect.2, we present the related works. The problem is described in Sect.3. In Sect.4, we define the dynamic communication area concept and its parts, used in the planner. We present TC-RRT to solve the time-constrained trajectory planning problem in Sect.5. In Sect.6, we discuss the results obtained by our method in simulations. Finally, in Sect.7, we present several conclusions and the future work.

2 Related work

The centralized approach for exploration missions is the most secure method to assure the coordination and the connectivity among the robots of the team. In works as [1] and [2], the teams are coordinated to accomplish a deployment mission. Both methods fully coordinate the team and are constrained to continuous connectivity between members, which in many cases might be a very strict constraint. We find interesting the works [3]-[6] in how they solve Time-Constrained planning. In [3], the authors avoid coordination, exploiting the signal of other members, but the algorithm is still restricted to a permanent communication. With our approach, the robot should be free to decide when to transmit the information and the period of this transmission. The communication time is relaxed in [4] and [5], but coordinating the robots is still needed specifically to this end. In [4], the team periodically establish *line-of-sight* communication, but does not take account of the time of the data exchange. In [5], a robot communicates with the team only when makes an observation, but it needs to be coordinated with the teammates. In [6], a multi-robot synchronization problem is addressed, where the robot trajectories are prefixed loops and the areas for inter-robot communication are also a priori established between each pair of trajectories. In our work, on the contrary to those previous works, the robots share information without disrupting the trajectories of the relays, and without pre-fixed communication areas between trajectories. Furthermore, the aforementioned techniques require a specific allocation algorithm for connectivity tasks. The presented trajectory planner avoids the usage of this specific task allocator for connectivity, because how and where to achieve the coordination is computed by the trajectory planner.

The trajectory computation involves the time in which the path is traversed, so the planner explores different temporary options to reach the goal. The runtime of methods such as Dijkstra, Fast Marching Method (FMM), or A* scale with the number of dimensions. Therefore, we consider sampling-based methods, which does not require the grid discretization. The randomized multiple-query algorithms such as Probabilistic Roadmaps (PRM) [7], are still computationally heavy for our problem. Therefore, we employ as the base method Rapidly-exploring Random Trees (RRT)[8]. Due to its single-query and randomized nature, it fits well to explore several trajectories, choosing the best one among them.

There are many variations of RRTs in the literature. The RRT-Connect [9] raises two trees, one from the initial position of the robot, and another from the goal. This way, the environment is explored faster and it reduces the probability to getting stuck. The Dynamic-Domain RRTs (DD-RRT) [10] limit the sampling domain, generating random nodes in areas which do not lead towards the obstacles and minimizing the execution time. The solution cost is improved by RRT* [11] and Informed RRT* [12]. RRT* improves the parent selection and includes a reconnection routine of neighboring nodes, reducing the cost of possible paths, with respect to the basic RRT. The Informed RRT* employs heuristics to delimit the sampling domain, finding out solutions faster and in more problematic scenarios where the randomized algorithms are not usually effective, as narrow passages. Both mentioned techniques increase the computation time with respect to the basic RRT.

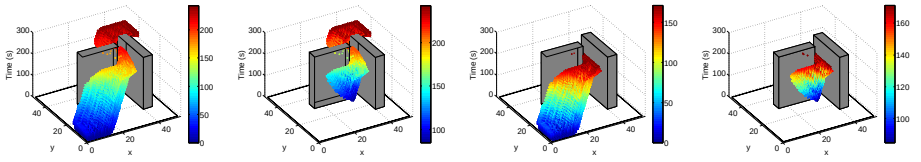
The presented TC-RRT, described in Sect.5, takes into account some of the features of the cited methods. We delimit the sampling space to lead the robot through the communication area, as occurs with DD-RRT and Informed RRT*. We use a parent selection routine, similar to RRT*, not to reduce the solution cost, but to avoid deadlock in a local minima, so increasing the success rate of the algorithm.

3 Problem setup

The problem what we solve is the computation of a trajectory to some location, which is synchronized with the trajectory of a relay robot, in order to exchange data. So this involves a spatio-temporal planning. Thus, let us define some location $\mathbf{x}$ as a duple of $[x\ y]^T$. As each position is visited at some time t, we can define a node $n = [\mathbf{x}\ t]^T$. Throughout the paper $\mathbf{x}(n)$ and $t(n)$ will denote position and time of some node n. Since the space is three-dimensional, the distance between a pair of nodes will include the difference between times, besides the Euclidean distance. Thus, the distance between any two nodes n_1 and n_2 is expressed as:

$$d(n_1, n_2) = \begin{bmatrix} \|\mathbf{x}(n_1) - \mathbf{x}(n_2)\| \\ |t(n_1) - t(n_2)| \end{bmatrix} \tag{1}$$

We denote τ as a trajectory travelled by a robot, that can be defined as a sequence of contiguous nodes: $\tau = [n_0, n_1, ...n_N]$, where N denotes the final



(a) $A(\tau_m)$ (b) A_{reach} (c) A_{ca} (d) A_{feas}

Fig. 2. Different communication area parts.

of the trajectory. The time of a trajectory to reach the goal is expressed with $t(\tau)$. The robot must communicate with a mate, which it is assigned for relaying the information to the operation center. The trajectory of the mate is expressed as τ_m, and it generates a dynamic communication area $A(\tau_m)$. The details of computation of this area are explained in Section 5. An example with a scenario in presence of obstacles is depicted in Fig.2(a). Knowing the size of the package to transmit, the robot determines the minimum time to fulfill the transmission, $t_{c_{min}}$. Then, the time that the trajectory of the robot is synchronized with τ_m, is expressed as $t_c(\tau)$. In summary, the problem to solve is to find the fastest trajectory which leads the robot to its primary goal, while it is synchronized the required time to accomplish the information transmission, Fig.1(d). Formally expressed as:

$$\begin{aligned} \tau^* = & \underset{\tau}{\operatorname{argmin}}(t(\tau)) \\ & \text{subject to} \quad t_c(\tau) \geq t_{c_{min}}. \end{aligned} \tag{2}$$

4 Dynamic communication area

A basic wireless antenna, located at some position $\mathbf{x}$ produces a communication area, denoted as $A(\mathbf{x})$. This area is composed by all the positions where the received signal strength is higher than some specified threshold, which assures the communication. Thus, it is possible to obtain the distance d_{th}, where the communication is guaranteed, from the following expression [14]:

$$P_{RX} = P_{TX} - 10\gamma log_{10}(d_{th}); \qquad d_{th} = 10^{\frac{P_{TX}-P_{RX}}{10\gamma}} \tag{3}$$

where P_{TX}, P_{RX} are the transmitted/received signal strength and γ is the path-loss exponent. Moreover, we want to assure always the communication, so we discard the positions which are obstructed by obstacles, considering only the *line-of-sight* (LoS) component. When a relay robot, with an antenna, travels its trajectory, it extends the static communication area with the movement. So, this dynamic area $A(\tau_m)$ can be expressed as all the nodes which are within the communication distance through the trajectory τ_m, Fig.2(a):

$$A(\tau_m) = \{n \mid \ dist(n_{m_i}, n) \wedge LoS(n_{m_i}, n), n_{m_i} \in \tau_m, i = 0, ..., N\} \tag{4}$$

where $dist(n_{m_i}, n)$ is a boolean function that computes $d(n_{m_i}, n) \leq [d_{th}\ \Delta t_{relay}]^T$ using eq.1, and Δt_{relay} is the maximum timestep of the relays trajectory. And $LoS(\tau_m, n)$ is a boolean function that computes if exists *line-of-sight* between the trajectory and the node, using the Bresenham algorithm [13].

The robot and the relay robot start from different initial positions. Thus, some nodes of $A(\tau_m)$ are impossible to reach and must be discarded. So, we define the reachable area by the robot, Fig.2(b), as:

$$A_{reach} = \{n \in A(\tau_m) \quad | \quad \|\mathbf{x}_0 - \mathbf{x}(n)\|/v_{max} \leq t(n)\} \tag{5}$$

where $\mathbf{x}_0$ is the initial position of the robot and v_{max} its maximum attainable speed.

As explained in the previous sections, the size of the package to transmit and the speed of the wireless link are known. So only those nodes of the communication area, which will guarantee the transmission of the data, are considered by the planner. So we define the communication assurance area A_{ca}, Fig.2(c), as:

$$A_{ca} = \{n \in A(\tau_m) \quad | \quad t(n) \leq t(n_N) - t_{c_{min}}\} \tag{6}$$

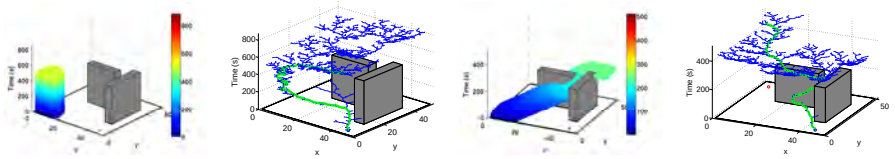
where $t(n_N)$ is the time of the last position of the trajectory of the mate τ_m. Therefore, the feasible area, Fig.2(d), which assures that the robot is able to reach the relay robot at time and guarantees a minimal time for the data exchange, is the intersection of the areas of eq.(5)-(6):

$$A_{feas} = A_{reach} \cap A_{ca} \tag{7}$$

If $\exists n_{feas} \in A_{feas}$, it means that the mission may be accomplished if the location of $\mathbf{x}(n_{feas})$ is reached no later than $t(n_{feas})$. However, if $A_{feas} = \emptyset$, there is no solution. TC-RRT guides the search to the goal through the communication area, in order to synchronize the robot with the relay.

5 Trajectory planning with limited communication time

The trajectory planner uses as base method the basic RRT. The proposed TC-RRT (Alg.1) has the same structure, but with different sampling and parent selection functions. First of all, let us define each node in the tree as $z = [\mathbf{x}\ p\ t\ a\ t_c]^T$, where $\mathbf{x}(z)$ and $p(z)$ are the position and the parent of z, $t(z)$ is the time to reach z, $a(z)$ is a boolean to indicate if z is within $A(\tau_m)$, and $t_c(z)$ is the communication time accumulated up to z in the tree. All these variables are computed when the node is inserted in the tree, with *InsertNode*. The algorithm generates random samples $\mathbf{x}$ in the workspace outside the obstacles (*AreaSample*), l.3 in Alg.1. The way to generate samples depends on if they are within or outside A_{feas}. This sample is connected to a parent in the tree by means of *AreaNearest* procedure (l.4), selecting this way the node z_{near} that provides the fastest movement. As in the basic *RRT*, the *Steer* function cut the stretch of the line between the generated sample ($\mathbf{x}_{sample}$) and the selected parent (z_{near}), l.5. If the new branch is not obstructed by some obstacle, the node



(a) (b) (c) (d)

Fig. 3. Tree (trajectory) computation for different scenarios. In (a), a robot that has collected data at [45,5] (red circle), must share information with some static mate at [5,5], then return to the initial position. In (b), the tree explores the environment, looking for the static communication area of the mate. When $A(\tau_m)$ is achieved, the robot remains within $A(\tau_m)$ until finish data exchange. Finally, $\mathcal{T}$ is expanded to the goal. The computed trajectory is depicted in green. In (c), the mate is moving, thus the area is dynamic. The robot must transmit data, before reaching the goal at [5,45]. In (d), $\mathcal{T}$ intercepts the communication trail area of the mate as fast as possible, remains in $A(\tau_m)$ until the end of communication, then achieves the primary goal.

is inserted in the tree $\mathcal{T}$, l.6-8. After expanding K nodes, there may be several branches or trajectories which achieve the goal. All of them fulfill the condition of the communication time, thus we select the fastest trajectory, as defined in eq.2. The algorithm iterates until expand the number of nodes specified by the user. This way, the environment is rapidly explored, although without the optimal solution, because of the randomized nature of the algorithm.

Two examples of tree computation and their respective trajectories are shown in Fig.3. Let us explain in more detail the algorithm.

InsertNode routine (Alg.2) receives the position of a new node and the parent, and computes the time to reach the node from the parents position (l.1), where v represents the velocity of the robot. Then it computes the total time to reach it from the root (l.2). It checks the presence in the area by means eq.1 in l.3, where d_ϵ represents a small value of distance. Note that this operation is quite lightweight, because Δt_{relay} selects only one horizontal slice of $A(\tau_m)$. If both,

Algorithm 1 TC-RRT

1: $\mathcal{T} \leftarrow$InsertNode($\mathbf{x}_{ini}, 0$)
2: **for** i=1:K **do**
3: $\quad \mathbf{x}_{sample} \leftarrow$ AreaSample
4: $\quad z_{near} \leftarrow$ AreaNearest($\mathbf{x}_{sample}$)
5: $\quad \mathbf{x}_{new} \leftarrow$ Steer($\mathbf{x}_{sample}, \mathbf{x}(z_{near})$)
6: $\quad$ **if** ObstacleFree($\mathbf{x}_{new}$) **then**
7: $\quad\quad \mathcal{T} \leftarrow$InsertNode($\mathbf{x}_{new}, z_{near}$)
8: $\quad$ **end if**
9: **end for**
10: **return** $\mathcal{T}$

Algorithm 2 InsertNode($\mathbf{x}_{new}, z_{near}$)

1: $\Delta t = \frac{\|\mathbf{x}_{new} - \mathbf{x}(z_{near})\|}{v(\mathbf{x}_{new}, \mathbf{x}(z_{near}))}$
2: $t_{new} = t(z_{near}) + \Delta t$
3: $a_{new} = \exists n \ : \ d([[\mathbf{x}_{new} \ t_{new}]^T, n]) \leq [d_\epsilon \ \Delta t_{relay}]^T, n \in A(\tau_m)$
4: **if** a_{new} & $a(z_{near})$ **then**
5: $\quad t_{c_{new}} = t_c(z_{near}) + \Delta t$
6: **else**
7: $\quad t_{c_{new}} = 0$
8: **end if**
9: $\mathcal{T} \leftarrow [\mathbf{x}_{new} \ z_{near} \ t_{new} \ a_{new} \ t_{c_{new}}]^T$

Algorithm 3 AreaSample

1: **if** $\exists z : \{dist([\mathbf{x}(z)\ t(z)]^T, n), n \in A_{feas} \wedge t_c(z) < t_{c_{min}}\}$ **then**
2: $z_a = \text{argmax}_z(t_c(z))$ ▷ Node which accumulates the maximal communication time
3: $\mathbf{x}_m^- = \mathbf{x} : \|\mathbf{x} - \mathbf{x}_m(t(z_a) + t_{min})\| \leq d_{th}$ $\mathbf{x}_m^+ = \mathbf{x} : \|\mathbf{x} - \mathbf{x}_m(t(z_a) + t_{max})\| \leq d_{th}$
4: $\mathbf{x}_{sample} \leftarrow rand(\mathbf{x}_m^-, \mathbf{x}_m^+)$
5: **else**
6: $\mathbf{x}_{sample} \leftarrow rand(\mathbf{x}_{min}, \mathbf{x}_{max})$ ▷ $\mathbf{x}_{min}, \mathbf{x}_{max}$ are the limits of the scenario
7: **end if**
8: **return** $\mathbf{x}_{sample}$

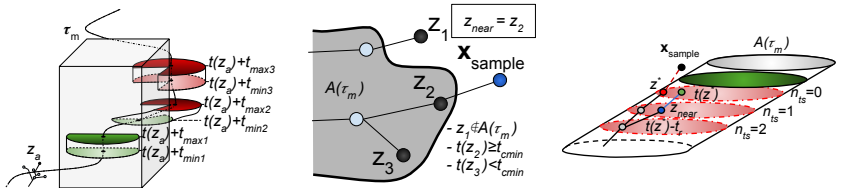
Algorithm 4 AreaNearest($\mathbf{x}_{sample}$)

1: **if** $\nexists z : a(z)$ **then**
2: $z_{near} = \text{argmin}_z \left(\frac{\|\mathbf{x}_{sample} - \mathbf{x}(z)\|}{v(\mathbf{x}_{sample}, \mathbf{x}(z))} \right)$
3: **else**
4: **if** $\exists z : t_c(z) \geq t_{c_{min}}$ **then**
5: $z_c = \{z \mid t_c(z) \geq t_{c_{min}}\}$
6: $z_{near} = \text{argmin}_{z \in z_c} \left(\frac{\|\mathbf{x}_{sample} - \mathbf{x}(z)\|}{v(\mathbf{x}_{sample}, \mathbf{x}(z))} \right)$
7: **else**
8: $z^* = \text{argmax}_z(t_c(z)) \ : \ a(z)$ ▷ Maximum communication time of the entire tree
9: $z_c = \{z \in z^* \mid t_c(z) \geq t_c(z^*) - t_r\}$
10: $z_{near} = \text{argmin}_{z \in z_c}(t(z))$
11: **end if**
12: **end if**
13: **return** z_{near}

the parent and the node, are within $A(\tau_m)$, the time of l.1 is accumulated as communication time (l.4-8).

AreaSample (Alg.3) works as follows. When some node is introduced into the communication area on time, i.e. in A_{feas} (l.1), and does not fulfill the condition of communication time of eq.(2) ($t_c(z) < t_{c_{min}}$), the tree is expanded through $A(\tau_m)$. The algorithm selects the node which accumulates the maximal communication time, z_a in l.2, and choose a greater time of the relays trajectory, using t_{min}, t_{max} in l.3, and delimiting the sampling space by the distance d_{th}. The selection of time limits is made in accordance with the relative speed between the relay and the robot, setting $t_{min} = \Delta t_{relay} v_{relay}/v_{robot}$ and $t_{max} = t_{min} + \Delta t_{relay}$, see Fig. 4(a). In the opposite case, if there exist no nodes which have entered within the communication area, A_{feas}, or the communication time has already been accumulated, so that $t_c(z) \geq t_{c_{min}}$, the samples are generated outside the obstacles in all the scenario, l.5-7.

AreaNearest (Alg.4) finds out the best parent node of the tree to connect the generated sample. The procedure is represented in Fig.4(b). If there are no nodes of the tree inside $A(\tau_m)$, l.1, it is connected to parents that provide the fastest movement, l.2. In the case that some node is within $A(\tau_m)$, l.3, and accomplishes the condition of eq.2, l.4, it selects all the nodes accomplishing that condition,



(a) Sampling area (b) Parent selection (c) Relaxation time

Fig. 4. Sampling area and nearest node selection. (a) shows the sampling space within communication area $A(\tau_m)$. The maximum communication time is achieved with the node z_a, green and red slices depict the visible and obstructed parts of $A(\tau_m)$, respectively. The obstructed parts produce deadlock situations, because the tree attempts to expand a branch which collides with an obstacle. (b) depicts the selection of suitable parent candidate. In this case, the best one is z_2, because it accomplishes the condition of minimal communication time, of eq.(2). In (c), $\mathbf{x}_{sample}$ is generated in the sampling space (green slice), but connecting to the best parent z^* (in red), the tree leaves $A(\tau_m)$. Using a relaxation time t_r, it connects to a suitable parent z_{near} (in blue), which keeps the tree within $A(\tau_m)$.

l.5, and chooses the parent of the fastest movement, l.6. When there are no nodes in $A(\tau_m)$ which accomplish $t_c(z) \geq t_{c_{min}}$, l.7, the suitable candidates to parent are those that have accumulated the maximum communication time of the entire tree, l.8. To increase the number of suitable candidate parents, a relaxation time t_r is applied in l.9. It is computed as $t_r = n_{ts} d_{max}/v_{max}$, where n_{ts} is the number of timesteps and d_{max} and v_{max} are the maximum attainable step and speed of the robot, respectively. Finally, the chosen parent is that provides the minimal time, l.10. Therefore, the selected parent is one that reduces the nodes time, maintaining the tree within $A(\tau_m)$, see Fig.4(c).

6 Simulations and discussion

In this section we discuss the performance of the TC-RRT and present its limitations. We test the method in the scenario of Fig.3(c), in which the robot has to capture a moving mate to transmit the collected data. The technique is evaluated for different system parameters. These parameters are the number of extended nodes, the communication time $t_{c_{min}}$ and the relative velocity between the robot and the relay. The communication time is selected as the percentage of the time available to transmit data, that is, the time of A_{reach}. We select: (1000, 2000, 5000, 10000) nodes, (25%, 50%, 75% and 90%) of time of A_{reach} and (-10%, 0%, +10%) for the speed of the robot with respect to the relays. Varying the relative speed, the slope of $A(\tau_m)$ changes. Increasing the speed of the relay or reducing the speed of the robot, the slope is reduced, which means, that it is more difficult to remain synchronized. The performance of the method is tested in terms of the solution cost, i.e. total time for the trajectory to the goal ($t(\tau)$), success rate and the execution time. The algorithm is implemented in MatLab and tested on a machine Intel Core i7 clocked at 3.40 Ghz and 8Gb

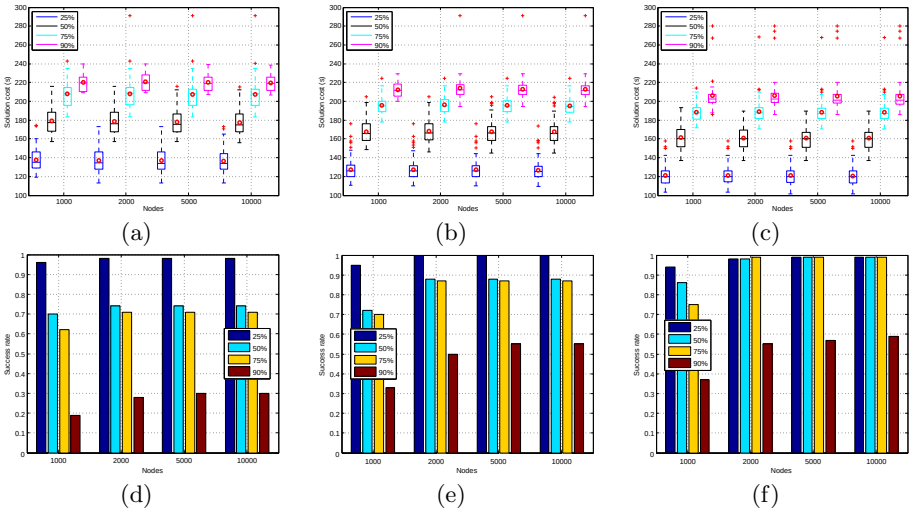


Fig. 5. Results for 100 random trials in the scenario of Fig.3(c). Each column represents -10%, 0% and +10% of relative speed, with respect to the speed of the mate. The red circles in (a)-(c) represent the mean value of the costs of the trajectories.

of memory. The results are presented in Fig.5 and an example of tree expansion for the tested scenario can be found in the video[1].

The cost of the solution does not change significantly with the number of nodes, Fig.5(a)-5(c). This is because the base method is the basic RRT, thus when some branch reaches A_{reach}, as well as the goal, it is unlikely that some other drastically different branch will reach the same positions. However, the success rate increases when some nodes threshold is exceeded, Fig.5(d)-5(f). This is clear in the transition from 1000 to 2000 nodes and for 90% of time of A_{reach}.

When the speed of the robot increases with respect to the relay, the success of the algorithm increases. Logically, the robot reaches the communication area and the goal faster, thereby reducing the solution cost. High values of $t_{c_{min}}$ reduce the time range to achieve the communication area on time. This, together with the randomized nature of the algorithm makes impossible to guarantee the communication. Even in the best case, 10% of extra speed, the success rate is just 60% for 90% of A_{reach} (equivalent to an interval of 8 seconds).

Expanding over 2000 nodes for this scenario, the solution is stabilized, the cost and success rate remain practically constant. The medium values of execution times were (1.3, 2.25, 5.47, 11.76) seconds for (1000, 2000, 5000, 10000) nodes respectively. Considering that for this scenario the results are good enough with few nodes (2000), it takes about 2 seconds to obtain quite steady solution.

As described in Sect.5, we employ a relaxation time t_r to increase the success rate of the algorithm. We set $n_{ts} = (0, 1, 2)$, and the results are depicted in

[1] http://robots.unizar.es/data/videos/robot17yamar.mp4

Table 1. Results for different timesteps, $n_{ts} = 0, 1, 2$.

	Execution time				**Success rate**				**Solution cost**			
n_{ts}	**1000**	**2000**	**5000**	**10000**	**1000**	**2000**	**5000**	**10000**	**1000**	**2000**	**5000**	**10000**
0	1.164	2.064	5.16	11.16	0.74	0.88	0.9	0.91	186.1	186.87	186.58	185.81
1	1.27	2.166	5.13	11.27	0.76	0.95	0.95	0.96	187.09	188.36	187.05	187.14
2	1.32	2.21	5.29	11.39	0.75	0.99	0.99	0.99	188.05	188.05	187.38	187.06

Table 1. The solution cost does not vary significantly, it is slightly lower without adding t_r. Logically, increasing n_{ts}, the number of evaluated candidates increases, but the execution time do not raises significantly. At the same time, using the relaxation factor increases the success rate, because the tree expansion avoids getting stuck within coverage area. This means that it is worthwhile to use this parameter. The simulations of Fig.5, were performed with $n_{ts} = 2$.

The method works under the hypothesis that the global plan for the robots will be maintained. If the scenario or the strength of the signal change, it might produce variations in the planned trajectories executed by the teammates. In this case, if the robot trajectories do no change a lot, only are deviated from the original trajectory, the technique could still work by extending the communication area to be explored around the original communication planned area. In case of this fails, another more costly solution can be that the mate returns to a previous communicated position in its trajectory, relaunching the planer. This is obviously a problem that will be formally dealt in future work.

7 Conclusions

In the present work, we have developed a trajectory planning algorithm, called TC-RRT, constrained by a communication time for exchange information. The planner computes a trajectory that guides the robot to some location, and at the same time synchronizes with another robot, just the time required to transmit the data. The presented technique is aimed to be used in scenarios where a team of robots is exploring an environment and some robots are used as relays. But unlike other solution in the literature, the relay robots are collecting data as well. So that, they retransmit the gathered data of their teammates to the OP when go to share its own data. This way, the coordination and permanent communication between the entire team is not required. Knowing the trajectory of the relay, each robot is able to plan the better place and instant to share its data. Likewise, the trajectory is obtained in accordance with the size of the data to share, so reduces substantially the total time for communication tasks.

The success rate of TC-RRT does not increase significantly with the number of nodes, exceeding a certain number of them. The reason is the usage of a basic RRT as the base of the method, it maintains the first result found and does not improve it. Therefore, an obvious future work is the adaptation to TC-RRT*, which includes an improvement in parent selection $AreaNearest$, as well as a rewiring routine to focus the tree construction within the communication area.

The sampling procedure and parent selection guide the trajectory within communication area of the relay. However, we will explore to employ a potential-like function to consider the displacement of the communication area, and compute better the possible movements within it.

References

1. J. Fink, A. Ribeiro and V. Kumar, Robust Control of Mobility and Communications in Autonomous Robot Teams, IEEE Access, pp. 290-309, vol.1, 2013.
2. D. Tardioli, D. Sicignano, L. Riazuelo, A. Romeo, JL. Villarroel and L. Montano, Robot Teams for Intervention in Confined and Structured Environments, Journal of Field Robotics, vol.33, no.6, pp.765-801, 2016.
3. E. F. Flushing, M. Kudelski, L. M. Gambardella and G. A. Di Caro, Spatial prediction of wireless links and its application to the path control of mobile robots, Proceedings of the 9th IEEE International Symposium on Industrial Embedded Systems (SIES 2014), pp. 218-227, ISSN. 2150-3109, June, 2014.
4. G. A. Hollinger and S. Singh, Multirobot Coordination With Periodic Connectivity: Theory and Experiments, IEEE Transactions on Robotics, vol.28, no.4, pp.967-973, 2012.
5. J. Banfi, A. Q. Li, N. Basilico, I. Rekleitis and F. Amigoni, Asynchronous multirobot exploration under recurrent connectivity constraints, IEEE International Conference on Robotics and Automation (ICRA), pp. 5491-5498, 2016.
6. J. M. Daz-Bez, L. E. Caraballo, M. A. Lopez, S. Bereg, I. Maza, A. Ollero, A General Framework for Synchronizing a Team of Robots Under Communication Constraints, in IEEE Transactions on Robotics , vol.PP, no.99, pp.1-8, 2017
7. N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. IEEE International Conference on Robototics and Automatioon, pp.113120, 1996.
8. S.M. LaValle, Rapidly-exploring random trees: A new tool for path planning, TR 98-11, Computer Science Dept., Iowa State University, Oct. 1998.
9. J. J. Kuffner and S. M. LaValle, RRT-connect: An efficient approach to single-query path planning, IEEE International Conference on Robotics and Automation, vol.2, pp.995-1001, 2000.
10. A. Yershova, L. Jaillet, T. Simeon, and S. LaValle, Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain, IEEE International Conference on Robotics and Automation, pp. 38563861, April 2005.
11. S. Karaman and E. Frazzoli, Sampling-based algorithms for optimal motion planning, The International Journal of Robotics Research, vol. 30, no. 7, pp. 846894, 2011.
12. J. D. Gammell, S. S. Srinivasa and T. D. Barfoot, Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic, IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 29997-3004, September, 2014.
13. Kenneth I. Joy, Breshenhams algorithm. In Visualization and Graphics Research Group, Department of Computer Science, University of California, Davis, 1999, Online: http://graphics.idav.ucdavis.edu/education/GraphicsNotes/Bresenhams-Algorithm.pdf.
14. J. Goldhirsh, W.J. Vogel, Handbook of Propagation Effects for Vehicular and Personal Mobile Satellite Systems: Overview of Experimental and Modelling Results, December, 1998